Context-dependent Sequential Recall by a Trajectory Attractor Network with Selective Desensitization

Masahiko MORITA [1), Kazuhiko MURATA [2), Shigemitsu MOROKAMI [2)

1) Institute of Engineering Mechanics and Systems, University of Tsukuba

2) Doctoral Program in System Information Engineering, University of Tsukuba

1-1-1 Tennodai, Tsukuba, Ibaraki 305-8573, Japan

E-mail: mor@bcl.esys.tsukuba.ac.jp

URL: http://volga.esys.tsukuba.ac.jp/~mor/

Abstract: *We present a model composed of nonmonotonic neurons that recalls various target patterns from the same cue pattern depending on a given context pattern, which was essentially difficult for conventional neural network models. In this model, the state of the network is projected onto a subspace by desensitizing a part of neurons depending on the context, and shifts along a trajectory attractor in the subspace to reach the target state. The model can simulate an arbitrary finite state automaton without limitations of size, keeping the merits of distributed representation.*

***Keywords*:** state transition, trajectory attractor, context-dependent association, nonmonotonic neuron, finite automaton.

1. INTRODUCTION

Let us consider a simplified problem of context-dependent association in a neural network with fully distributed representation. Specifically, target pattern $T^{\mu\nu}$ to be recalled depends not only on cue pattern S^{μ} $(\mu = 1, \cdots, p)$ but also on pattern C^{ν} $(\nu = 1, \cdots, q)$ representing context, where the elements of patterns S^{μ}, C^{ν} and $T^{\mu,\nu}$ take 1 or -1 randomly with equal probabilities. Although this problem appears very simple, conventional neural networks have serious difficulty in solving it.

For example, a two-layer network shown in Fig.l (a) in which the input units representing S^{μ} and C^{ν} are directly connected to the output units cannot realize the input-output relation expressed by Table 1. This is because the set of target patterns are the same for every S^{μ} and for every C^{ν}. Even if all target patterns are mutually different, no connection weights solve the problem when *p* and *q* are large enough, since the average of many target patterns associated with the same S^{μ} (or C^{ν}) does not vary much with μ (ν) because of the averaging effect. Moreover, this difficulty is not essentially resolved by introducing hidden units, between the input and output units unless we use local representation, *pq* hidden units each of which encodes a particular combination of S^{μ} and C^{ν}.

This kind of contextual modification such that $S(C)$, an object S in a context C, is represented by the concatenated pattern (S, C) has been used in most models of contextual processing. However, they do not work well unless the size (*p* and *q*) of the problem is small enough, due to the above difficulty caused by one-to-many correspondence. This means that contextual processing by neural networks based on fully distributed representation has strong restriction.

This paper presents a model of context-dependent association that solves the above problem without using local representation.

2. THE MODEL

2.1. Dynamics and contextual modification

The present model has been developed by modifying the dynamics of the analog Hopfield model, namely continuous-time fully recurrent neural networks. Specifically, the *i*-th neuron $(i = 1, \cdots, n)$ acts according to

$$\tau \frac{du_i}{dt} = -u_i + \sum_{j=1}^{n} w_{ij} y_j + z_i \ , \tag{1}$$

$$y_i = g_i \cdot f(u_i) \ , \tag{2}$$

where u_i denotes the instantaneous potential, w_{ij} the synaptic weight from the *j*-th neuron, y_i the output, z_i the external input, τ a time constant, g_i output gain, and $f(u)$ the activation function. These equations are the same as conventionally used except that g_i is controllable and $f(u)$ is a nonmonotonic function

$$f(u) = \frac{1 - e^{-cu}}{1 + e^{-cu}} \cdot \frac{1 - e^{c'(|u| - h)}}{1 + e^{c'(|u| - h)}} \ , \tag{3}$$

where *c*, *c*' and *h* are positive constants (we substitute *c*=50, *c*'=10, *h*=0.5 in the experiments described later).

Since the polarity of u_i is important in this model, we consider $x_i = \mathrm{sgn}(u_i)$ and treat the vector $\boldsymbol{x} = (x_1, \cdots, x_n)$ as the network state, where $\mathrm{sgn}(u) = 1$ for $u > 0$ and -1 for $u \leq 0$. The network state $\boldsymbol{x}$ at an instant is represented by a point in the state space consisting of 2^n possible states. When $\boldsymbol{x}$ changes, it almost always moves to an adjacent point in the state space because x_i changes asynchronously. Consequently, $\boldsymbol{x}$ leaves a track with time, which we call the trajectory of $\boldsymbol{x}$.

If $g_i \equiv 1$ for all neurons, the network is identical to an existing model known as the nonmonotone neural network (NNN[1]). In the present model, however, the gain g_i is

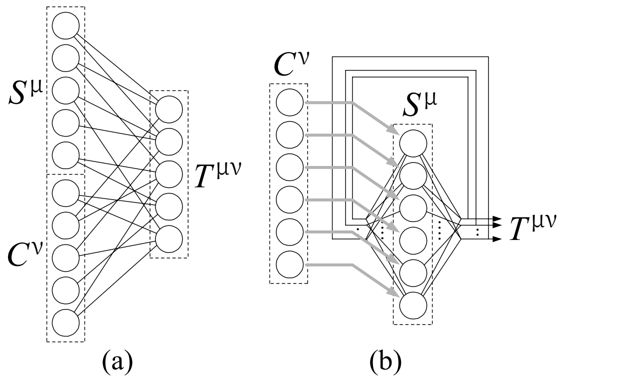

Fig.1 – Methods of contextual modification: (a) conventional concatenation, and (b) selective desensitization used in the present model.

Table 1. Example of the relation of target patterns to cue and context patterns that is difficult to realize.

	C^1	C^2	C^3	C^4	C^5	C^6	C^7	C^8	C^9	C^{10}
S^1	T^1	T^2	T^3	T^4	T^5	T^6	T^7	T^8	T^9	T^{10}
S^2	T^{10}	T^1	T^2	T^3	T^4	T^5	T^6	T^7	T^8	T^9
S^3	T^9	T^{10}	T^1	T^2	T^3	T^4	T^5	T^6	T^7	T^8
S^4	T^8	T^9	T^{10}	T^1	T^2	T^3	T^4	T^5	T^6	T^7
S^5	T^7	T^8	T^9	T^{10}	T^1	T^2	T^3	T^4	T^5	T^6
S^6	T^6	T^7	T^8	T^9	T^{10}	T^1	T^2	T^3	T^4	T^5
S^7	T^5	T^6	T^7	T^8	T^9	T^{10}	T^1	T^2	T^3	T^4
S^8	T^4	T^5	T^6	T^7	T^8	T^9	T^{10}	T^1	T^2	T^3
S^9	T^3	T^4	T^5	T^6	T^7	T^8	T^9	T^{10}	T^1	T^2
S^{10}	T^2	T^3	T^4	T^5	T^6	T^7	T^8	T^9	T^{10}	T^1

set to zero depending on the context. Since the output y_i of the neuron with $g_i = 0$ does not vary with the input, we refer to this operation as desensitization, and about half of the neurons are `desensitized' when the network receives a contextual modification. The gain vector $\boldsymbol{G} = (g_1, \cdots, g_n)$ is specified by the context and is different in different contexts, in other words, $\boldsymbol{G}$ represents the context.

In the following, we regard $\boldsymbol{G}$ in the same light as the context pattern C, and deal with the simplest case that $g_i = (1 + c_i)/2$, where c_i $(i = 1, \cdots, n)$ is the *i*-th component of C and takes ± 1. We also define the network state modified by context C as $\boldsymbol{x}(C) \equiv (g_1 x_1, \cdots, g_n x_n)$, which is a three-valued (1, 0, -1) vector. In the same way, we define the patterns S and T modified by C as $S(C) \equiv (g_1 s_1, \cdots, g_n s_n)$ and , $T(C) \equiv (g_1 t_1, \cdots, g_n t_n)$respectively.

2.2. Forming trajectory attractors

A NNN can have many string-shaped attractors, termed trajectory attractors, in its state space, along which the network stably makes continuous state transitions [2]. This enables the network to associate a cue with an arbitrary target, but not to recall different targets depending on contexts. Thus in the present model, we form trajectory attractors in the subspace composed of the valid (not desensitized) neurons in each context. The concrete algorithm for it is as follows.

First, we give an initial state $\boldsymbol{x}(C^{\nu}) = S^{\mu}(C^{\nu})$ and input the learning signal $\boldsymbol{r} = S^{\mu}(C^{\nu})$ in the form $z_i = \lambda r_i$ to the network modified by C^{ν}, where λ denotes the input intensity and r_i is the *i*-th element of $\boldsymbol{r}$.Then we change $\boldsymbol{r}$ bit by bit to $T^{\mu,\nu}(C^{\nu})$ over some period (5 τ in the simulations below), modifying synaptic weights w_{ij} between valid neurons according to

$$\tau' \frac{dw_{ij}}{dt} = -w_{ij} + \alpha \, r_i \, y_j \,, \qquad (4)$$

where τ' denotes a time constant of learning ($\tau' >> \tau$) and α is a learning coefficient (since performance of learning is better when α is a decreasing function of $|\,u_i\,|$ [2], we substitute $\alpha = 2.0 \, x_i \, y_i$ in the simulations).

While $\boldsymbol{r}$ is moving in the state space, $\boldsymbol{x}(C^{\nu})$ follows slightly behind. When $\boldsymbol{r}$ reaches the end $T^{\mu,\nu}(C^{\nu})$, we keep $\boldsymbol{r} = T^{\mu,\nu}(C^{\nu})$ for a while (1 τ) so that $\boldsymbol{x}(C^{\nu})$ may sufficiently approach $\boldsymbol{r}$.

In addition, we input $\boldsymbol{r} = T^{\mu,\nu}$ for a short time to the network without contextual modification ($\boldsymbol{G}=(1,\cdots,1)$) so that the state $\boldsymbol{x}=T^{\mu,\nu}$ may become a point attractor (this operation is unnecessary when $T^{\mu,\nu}$ naturally becomes an attractor, which is the case with the simulations below).

We apply the above procedure to all μ and ν, and repeat it over some (about 10 to 30) cycles, gradually decreasing λ. If $\boldsymbol{x}(C^{\nu})$ follows $\boldsymbol{r}$ even when λ = 0, then trajectory attractors have been formed.

2.3. Context-dependent recall

The process of recall is schematically shown in Fig. 2 where the *n*-dimensional state space of the network is expressed three dimensionally.

For example, when the network is modified by a context pattern C^1, the network state $\boldsymbol{x}$ is projected onto a subspace and it becomes $\boldsymbol{x}(C^1) = S(C^1)$.Then $\boldsymbol{x}(C^{\nu})$ moves along the trajectory attractor to $T^1(C^1)$, when it is expected that $\boldsymbol{x} = T^1$ because T^1 is an attractor (note that generally the potential u_i of desensitized neurons is not zero).Even if $\boldsymbol{x} \neq T^1$, the exact pattern T^1 is recalled when we release all neurons from desensitization. In the same way, if the state $\boldsymbol{x} = S$ receives modification by C^2 and C^3, the network recalls T^2 and T^3, respectively, by the state transitions along the trajectory attractors in respective subspaces.

It should be noted that Fig. 2 is somewhat misleading in that the projection directions of $\boldsymbol{x}$ by C^1, C^2 and C^3 are orthogonal, indicating no overlap of desensitized neurons; but in fact, on average, half of the desensitized neurons are common to two different contexts. It should also be noted that the output y_i of desensitized neurons should be at the mean level $\bar{y}$ (estimated to be 0 in this model) to minimize interference between different subspaces.

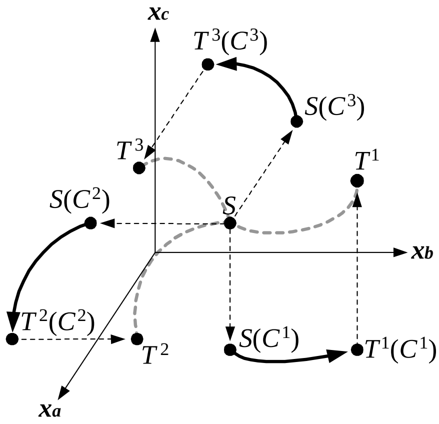

Fig.2 – Recall process in the state space.

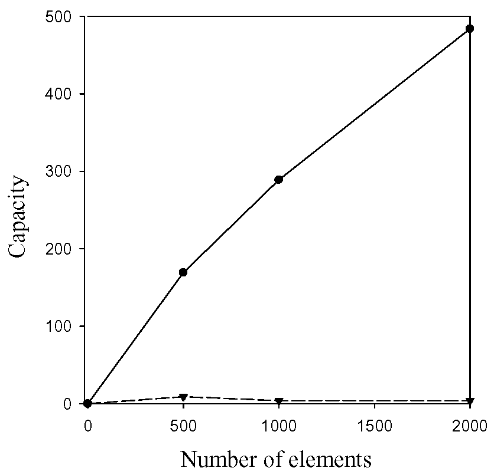

Fig.3 – Association capacity of the model.

3. COMPUTER SIMULATIONS

3.1. Capacity

We carried out computer simulations to examine the performance of the model. Fig. 3 shows one of the results, in which the maximum number *pq* of associations that the model can correctly form is plotted against the number *n* of neurons. In this experiment, cue patterns $S^1,\cdots,S^p$ and context patterns $C^1,\cdots,C^q$ are of the same number (*p*=*q*) and generated at random; the target pattern $T^{\mu,\nu}$ is one of the patterns $T^1,\cdots,T^p$ generated at random, and is determined by the remainder for $|\mu-\nu|$ divided by p as shown in Table 1.

As we see from the graph, the capacity of this model increases nearly proportionally with *n*. In contrast, when we use the conventional method of contextual modification (concatenating C to S), the capacity is almost zero even with a NNN, as shown by broken lines in Fig. 3.

We tested the model on various other conditions including the case $p \neq q$ and the capacity was larger than 0.16*n* in any case, indicating that this model is free from the above difficulty in context-dependent association. This is thought to be because neither S^μ or C^ν is directly associated with $T^{\mu,\nu}$.

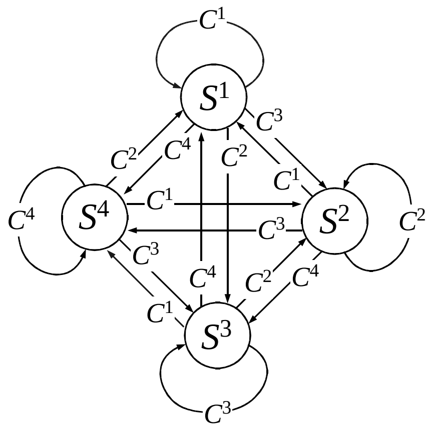

Fig.4 – Example of the state transition diagram.

3.2. Simulating a finite automaton

In our model, the target pattern may be identical to one of the cue patterns. Then another target pattern can be recalled from the recalled pattern, and by repeating this, target patterns can be sequentially recalled. In this process, we can control the sequence of patterns is also possible by switching the current context. In other words, the state transition of the network is controllable.

For example, any state transition among $S^1,\cdots,S^p$ is possible using *p* context patterns, as shown in Fig. 4 for the case of *p*=4. Also Table 2 shows an example of S^μ-C^ν relation that enables arbitrary state transitions in the case *p*=*q*=10.

In practice, we trained the network to act according to Table 2. Fig. 5 shows the behavior of the model when S^μ is not externally given except for the initial state, where similarities $\sum x_i s_i^\mu / n$ of the network state $\boldsymbol{x}$ to patterns S^1 to S^{10} are plotted against time scaled by the time constant τ.

We see in (a) that $\boldsymbol{x}$ shifts from the initial state S^6 to S^5, moving via S^7, S^4, S^8, etc. to S^1 while $C = C^1$ is kept throughout. On the other hand, $\boldsymbol{x}$ makes different transitions by switching C in (b). It should be noted that switch of C does not have to be made when $\boldsymbol{x} = S^\mu$ exactly holds; that is, we may switch C in the middle of state transitions except when $\boldsymbol{x}$ is midway between S^μ , as shown in (b) at t=38τ.

As we see from this example, the model can recall S^μ in any order and preserve S^μ for any time. This means that the model simulates a finite state automaton (FSA) with *p* states and *q* inputs, as is apparent from Fig. 4. Since the number of trajectory attractors or transition paths that the model can form increases proportionally with *n*, it follows that this model possesses the ability to simulate an arbitrary FSA of any size without explosion of neurons.

Table 2. State transition rules used for the experiment.

	C^1	C^2	C^3	C^4	C^5	C^6	C^7	C^8	C^9	C^{10}
S^1	S^1	S^2	S^3	S^4	S^5	S^6	S^7	S^8	S^9	S^{10}
S^2	S^{10}	S^2	S^1	S^3	S^4	S^5	S^6	S^7	S^8	S^9
S^3	S^9	S^{10}	S^3	S^1	S^2	S^4	S^5	S^6	S^7	S^8
S^4	S^8	S^9	S^{10}	S^4	S^1	S^2	S^3	S^5	S^6	S^7
S^5	S^7	S^8	S^9	S^{10}	S^5	S^1	S^2	S^3	S^4	S^6
S^6	S^5	S^7	S^8	S^9	S^{10}	S^6	S^1	S^2	S^3	S^4
S^7	S^4	S^5	S^6	S^8	S^9	S^{10}	S^7	S^1	S^2	S^3
S^8	S^3	S^4	S^5	S^6	S^7	S^9	S^{10}	S^8	S^1	S^2
S^9	S^2	S^3	S^4	S^5	S^6	S^7	S^8	S^{10}	S^9	S^1
S^{10}	S^1	S^6	S^7	S^2	S^8	S^3	S^9	S^4	S^5	S^{10}

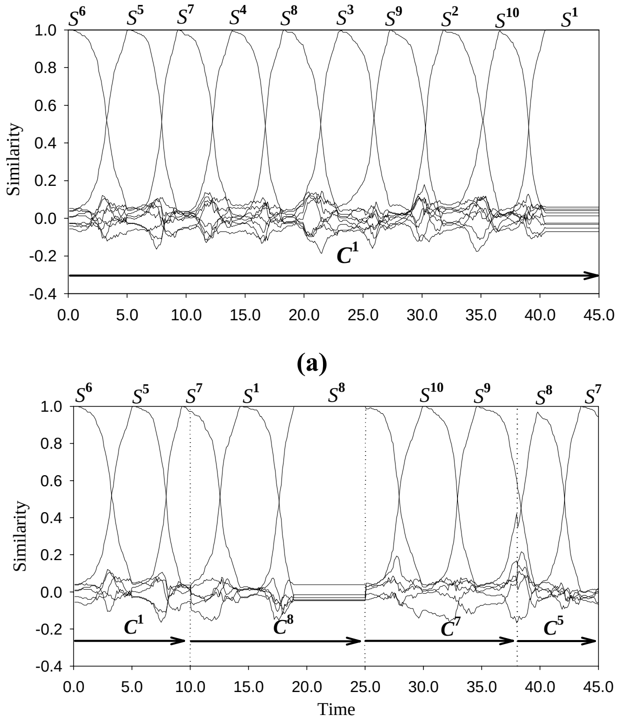

Fig.5 – Behavior of the model.

4. CONCLUDING REMARKS

Although the present model can simulate any FSA, it is not equivalent to a FSA, since the model is based on fully distributed representation and has high ability of generalization. For example, the model can receive an unknown combination of cue and context patterns and recall a proper target pattern by analogy. This implies that the model has, in principle, a potential for surpassing conventional intelligent systems based on symbolic manipulation.

To examine this possibility, however, further investigation is required, using not random but structured patterns.

We also point out that the principle of our model is biologically plausible. First, trajectory attractors can be formed in a network composed of pairs of excitatory and inhibitory neurons with a sigmoid activation function [3], and it is suggested that recall in the inferior temporal cortex is performed by a pattern shift in the neuronal activity [4,5] like the state transition along the trajectory attractor. Second, there exists good physiological evidence supporting that selective desensitization of neurons is used for contextual processing in the brain, but further discussion on this subject will be given at some other time.

5. REFERENCES

[1] M. Morita, "Memory and learning of sequential patterns by nonmonotone neural networks," *Neural Networks*, vol.9, no.8, pp.1477-1489, 1996.

[2] M. Morita, "Associative memory with nonmonotone dynamics," *Neural Networks*, vol.6, no.1, pp.115-126, 1993.

[3] A. Suemitsu, S. Morokami, K. Murata, and M. Morita, "Computational examination on the dynamics of recall activity in the inferior temporal cortex," *Proc.IJCNN 2002*, pp.136-141, 2002.

[4] M. Morita, and A. Suemitsu, "Computational modeling of pair-association memory in inferior temporal cortex," *Cognitive Brain Research*, vol.13, pp.169-178, 2002.

[5] M. Morita, "Computational study on the neural mechanism of sequential pattern memory," *Cognitive Brain Research*, vol.5, pp.137-146, 1996.